CS A262: DISCRETE STRUCTURES

Item	Value
Curriculum Committee Approval Date	04/08/2020
Top Code	070600 - Computer Science (Transfer)
Units	3 Total Units
Hours	90 Total Hours (Lecture Hours 36; Lab Hours 54)
Total Outside of Class Hours	0
Course Credit Status	Credit: Degree Applicable (D)
Material Fee	No
Basic Skills	Not Basic Skills (N)
Repeatable	No
Grading Policy	Standard Letter (S)
Associate Arts Local General Education (GE)	• OC Comm/Analytical Thinking - AA (OA2)
Associate Science Local General Education (GE)	• OCC Comm/AnalyticalThinking-AS (OAS2) • OCC Mathematics (OMTH)
California State University General Education Breadth (CSU GE-Breadth)	• CSU B4 Math/Quant.Reasoning (B4)

Course Description

An introduction to the discrete structures used in Computer Science with an emphasis on their applications. Topics covered include functions, relations, sets, basic logic, proof techniques, basics of counting, graphs, trees, and discrete probability. PREREQUISITE: CS A100, CS A122, CS A131, CS A150, or CS A170. Transfer Credit: CSU; UC. C-ID: COMP 152.**C-ID:** COMP 152.

Course Level Student Learning Outcome(s)

1. Recognize and use basic logic notation.
2. Demonstrate different traversal methods for trees and graphs.
3. Analyze recursive algorithms.

Course Objectives

- 1. Describe how formal tools of symbolic logic are used to model real-life situations, including those arising in computing contexts such as program correctness, database queries, and algorithms.
- 2. Relate the ideas of mathematical induction to recursion and recursively defined structures.
- 3. Analyze a problem to create relevant recurrence equations.
- 4. Demonstrate different traversal methods for trees and graphs.
- 5. Apply the binomial theorem to independent events and Bayes theorem to dependent events.

Lecture Content

Logic Propositional logic Logical connectives Truth tables Normal forms (conjuctive and disjunctive) Propositional equivalences De Morgans Laws Predicate logic Universal and existential quantification Validity Modus ponens and modus tollens Limitations of predicate logic Proofs Notions of implication, converse, inverse, contrapositive, negation, and contradiction The structure of mathematical proofs Direct proofs Proofs by counterexample Proofs by contradiction Functions One-to-one and onto functions (injective and surjective) Inverse functions Compositions of functions Sets Venn diagrams Cartesian products Power sets Complements Unions and intersections Sequences and Summations Arithmetic and geometric progressions Summation notation Cardinality Matrices Matrix arithmetic Transposes and power of matrices Zero-one matrices Algorithms The growth of functions Complexity of algorithms Sequential vs. binary search algorithms Relations Reflexive, symmetric, and transitive relations Equivalence relations Recurrence relations Solving linear recurrence relations Integers Representation of integers Division Modular arithmetic Primes and greatest common divisors Induction and Recursion Mathematical induction Strong induction Well ordering Recursive mathematical definitions Fibonacci Numbers Counting Counting arguments Sum and product rule Generating functions Inclusion-exclusion principle The pigeonhole principle Permutations and combinations The Binomial Theorem Pascals identity The Master Theorem Discrete Probability Finite probability The probability of combinations and events Conditional probability Independence Random variables Bayes Theorem Graphs and Trees Graph terminology Directed graphs Undirected graphs Spanning trees/forests Traversal strategies Expected values and variance Law of large numbers

Lab Content

The following is a list of possible programming labs that are related to the content topics: Logic Design and implement decision-making algorithms. Given truth values of propositions, find the truth values of the conjunction, disjunction, exclusive or, conditional statement, and bi-conditional of these propositions. Algorithms and Functions Decompose a program into functions that perform subtasks. Analyze and trace the execution of computer programs. Sets Write applications that determine the union and intersection of sets. Given a finite set, compute all elements of its power set. Design and implement an algorithm that returns the Cartesian product of two sets. Summations Implement loops to solve summations. Determine the number of comparisons used by different sorting algorithms. Matrices Represent matrices using two-dimensional arrays. Write applications that add and multiply matrices. Integers Implement Euclids algorithm for finding GCD. Write applications that convert numbers into binary, hexadecimal and/or decimal representations. Counting Create possible combinations and permutations using loops. Count the number of possible combinations and permutations through enumeration, for moderate number of possibilities. Recursion Find solutions to problems using recursive functions. Compare iterative solutions to recursive solutions. Graphs Represent graphs as adjacency matrices using arrays.

Method(s) of Instruction

- Lecture (02)
- DE Online Lecture (02X)
- Lab (04)
- DE Online Lab (04X)

Instructional Techniques

Lecture, demonstrations, exercises, and programming labs.

Reading Assignments

Students will spend a minimum of 4 hours per week reading the textbook and/or other reading material assigned. Students will be expected to follow along with the exercises in the reading material.

Writing Assignments

Students will spend a minimum of 6 hours per week writing logical, step-by-step solutions to assigned exercises and programming labs.

Out-of-class Assignments

Students will spend a minimum of 6 hours per week completing weekly programming assignments.

Demonstration of Critical Thinking

Critical thinking will be evaluated in both the midterm and final exams through a problem-solving approach.

Required Writing, Problem Solving, Skills Demonstration

Writing is encouraged throughout the course, but is not necessarily a part of the grading or the exams.

Eligible Disciplines

Computer science: Masters degree in computer science or computer engineering OR bachelors degree in either of the above AND masters degree in mathematics, cybernetics, business administration, accounting or engineering OR bachelors degree in engineering AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in mathematics AND masters degree in cybernetics, engineering mathematics, or business administration OR bachelors degree in any of the above AND a masters degree in information science, computer information systems, or information systems OR the equivalent. Note: Courses in the use of computer programs for application to a particular discipline may be classified, for the minimum qualification purposes, under the discipline of the application. Masters degree required.

Textbooks Resources

1. Required Irani, S., Edgcomb, A., Lysecky S., Vahid, F., Sui, R., Majidi, AJ. CS A262: Discrete Structures, 1st ed. Wiley, 2019